Atelier Data Science

Deep learning practice 4

Batch Normalization in DL

Irina Proskurina

Irina.Proskurina@univ-lyon2.fr

Laboratoire ERIC – Université Lyon 2

Mini-batch GD

1. Initial approximation w^0
2. Repeat, each time choosing a set of *n* random objects :

$$w^t = w^{t-1} - \eta_t \frac{1}{n} \sum_{j=1}^{n} \nabla L(y_{t,j}, a(x_{t,j}))$$

3. Stop when the error on the test set stops decreasing

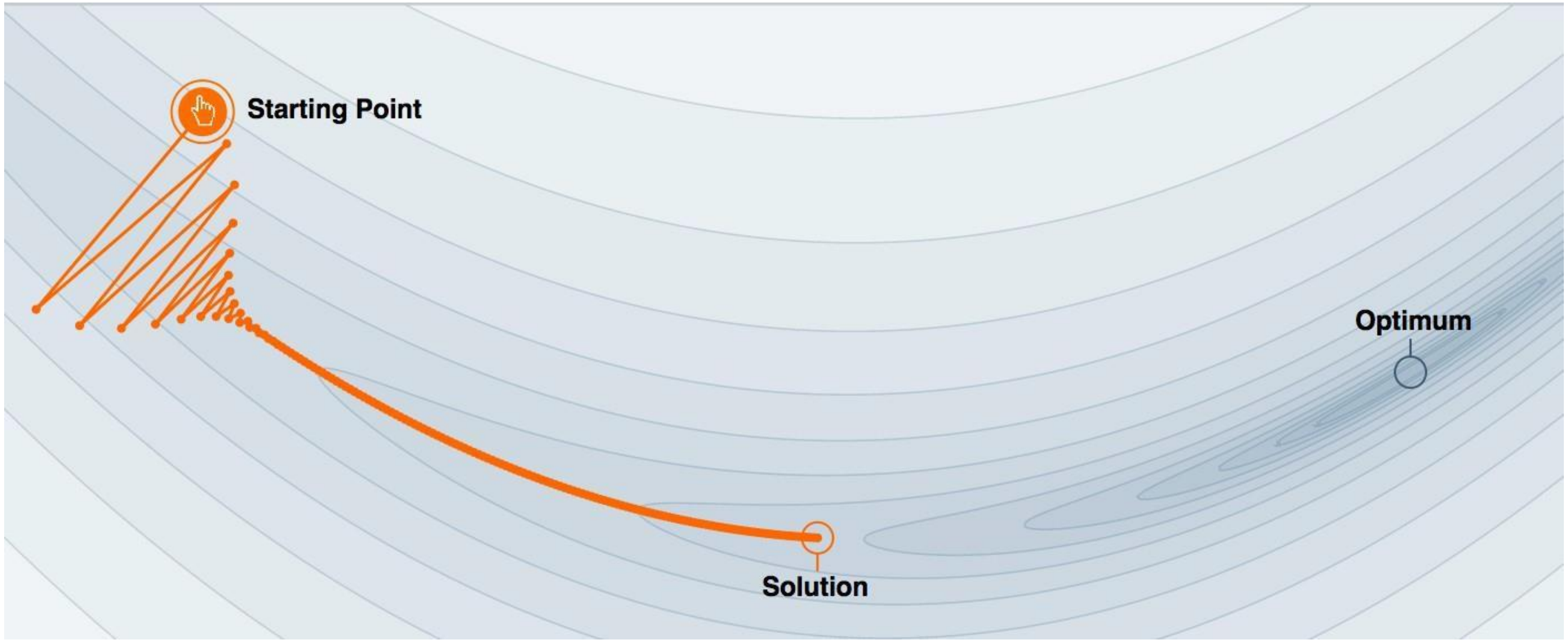
Momentum (smooths out the zig-zag path)

$$h_t = \alpha h_{t-1} + \eta_t \nabla Q(w^{t-1})$$

$$w^t = w^{t-1} - h_t$$

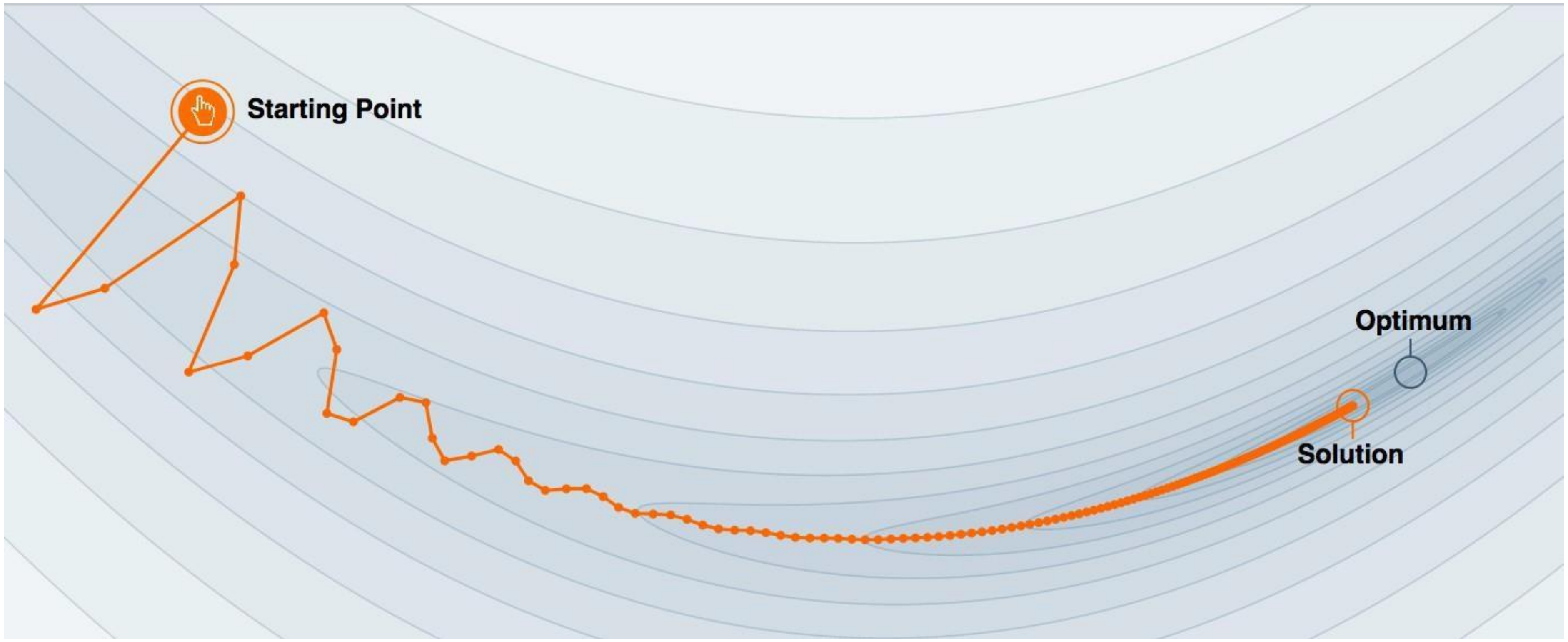
- h_t— is the velocity at time *t-1*
- α — momentum term

Without momentum



https://distill.pub/2017/momentum/

Momentum



https://distill.pub/2017/momentum/

AdaGrad

$$G_j^t = G_j^{t-1} + (\nabla Q(w^{t-1}))_j^2$$

$$w_j^t = w_j^{t-1} - \frac{\eta_t}{\sqrt{G_j^t + \epsilon}} (\nabla Q(w^{t-1}))_j$$

- For each parameter, there is its own learning rate can be fixed
- The step size may decrease too rapidly and lead to early stopping

RMSProp

$$G_j^t = \alpha G_j^{t-1} + (1-\alpha)(\nabla Q(w^{t-1}))_j^2$$

$$w_j^t = w_j^{t-1} - \frac{\eta_t}{\sqrt{G_j^t + \epsilon}} G_j^t$$

- $\alpha \sim$ 0.9
- The speed update depends only on recent steps

Adam

- Momentum + RMSProp

$$m_j^t = \frac{\beta_1 m_j^{t-1} + (1-\beta_1)(\nabla Q(w^{t-1}))_j}{1-\beta_1^t}$$

$$v_j^t = \frac{\beta_2 v_j^{t-1} + (1-\beta_2)(\nabla Q(w^{t-1}))_j^2}{1-\beta_2^t}$$

$$w_j^t = w_j^{t-1} - \frac{\eta_t}{\sqrt{v_j^t + \epsilon}} m_j^t$$

- Recommended values: $\beta_1 = 0.9$, $\beta_2 = 0.999$ $\epsilon = 10^{-8}$

Dropout

- Drop $d(x)$
- There are no parameters; the only hyperparameter is p (the probability of dropping a neuron). During the training phase:

$$d(x) = \frac{1}{p} m \circ x$$

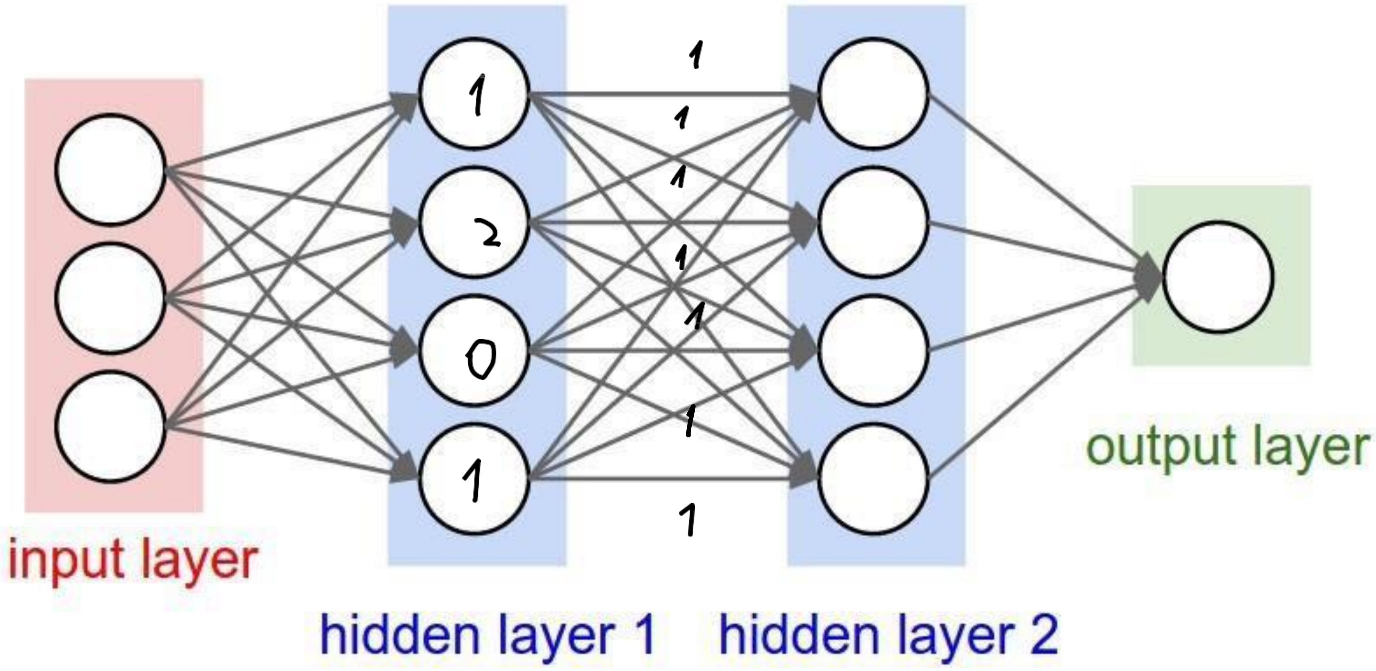
- m is a vector of the same size as x;
  elements are sampled from the Bernoulli distribution Ber(p)
- Division by p is done to preserve the overall scale of the outputs

https://jmlr.org/papers/v15/srivastava14a.html

BatchNorm

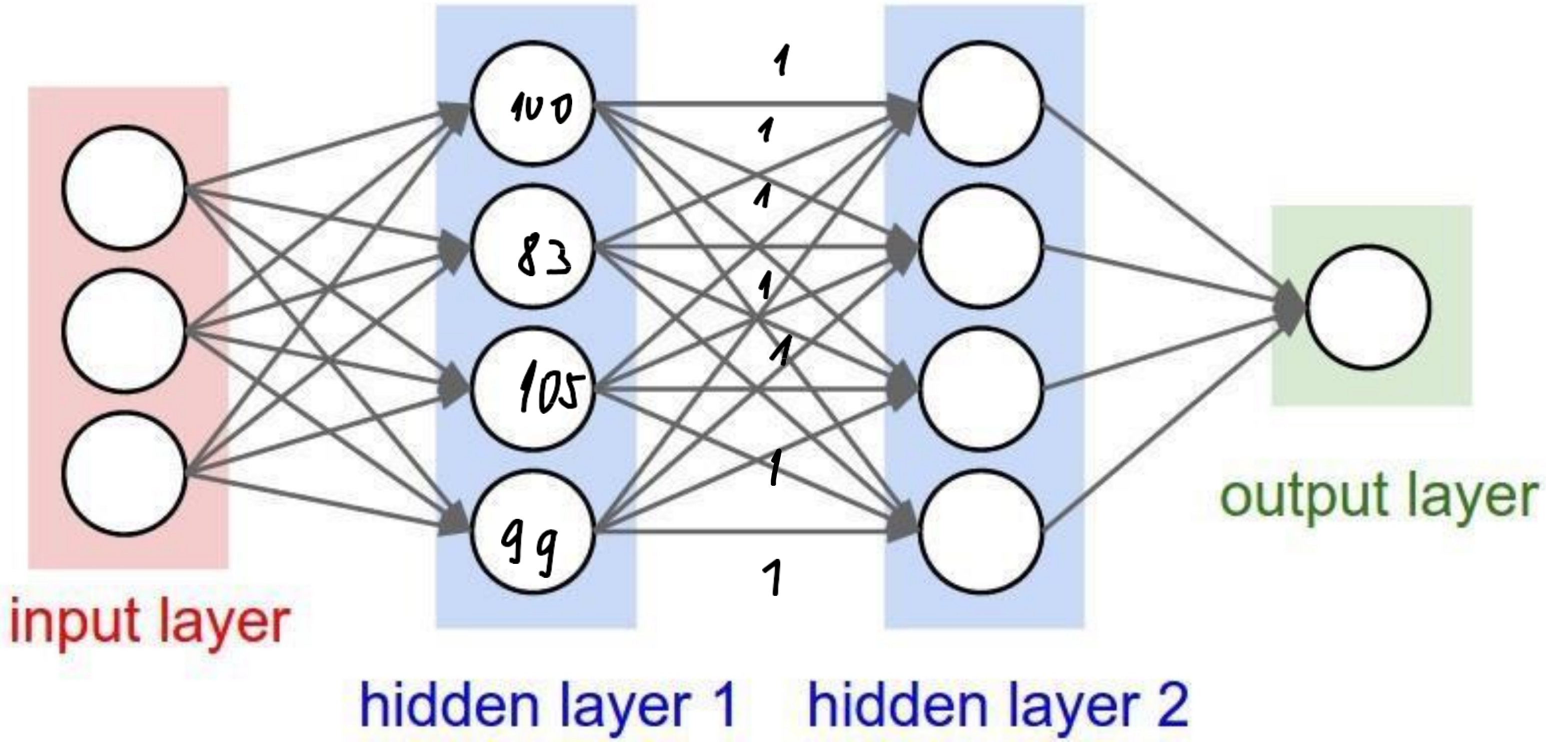
Internal covariate shift

- In a neural network, each layer's input is the output of the previous layer
- If a layer at the beginning undergoes significant changes, all subsequent layers need to be adjusted



Internal covariate shift

- Idea: Transforming the outputs of layers to ensure they consistently have a fixed distribution
- BatchNorm



Batch Normalization

- Implemented as a separate layer
- Computed for the processing batch
- Estimate the mean and variance for each (layer's) input vector

$$\mu_B = \frac{1}{n}\sum_{j=1}^{n} x_{B,j}$$

$$\sigma_B^2 = \frac{1}{n}\sum_{j=1}^{n} (x_{B,j} - \mu_B)^2$$

Batch Normalization

- Scale all layer outputs:

$$\tilde{x}_{B,j} = \frac{x_{B,j} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

- Scale with trained parameters, mean and variance:

$$z_{B,j} = \gamma \circ \tilde{x}_{B,j} + \beta$$

Batch Normalization

Important: after BatchNorm layer, the mean and variance of each output depend only on the normalization parameters, not on the parameters of previous layers!

- Usually inserted between a fully connected/convolutional layer and non-linearity
- Allows for an increase in the learning rate in gradient descent

- Not guaranteed to truly eliminate covariance shift

Why use BatchNorm?

How Does Batch Normalization Help Optimization?

Shibani Santurkar* MIT shibani@mit.edu

Dimitris Tsipras* MIT tsipras@mit.edu

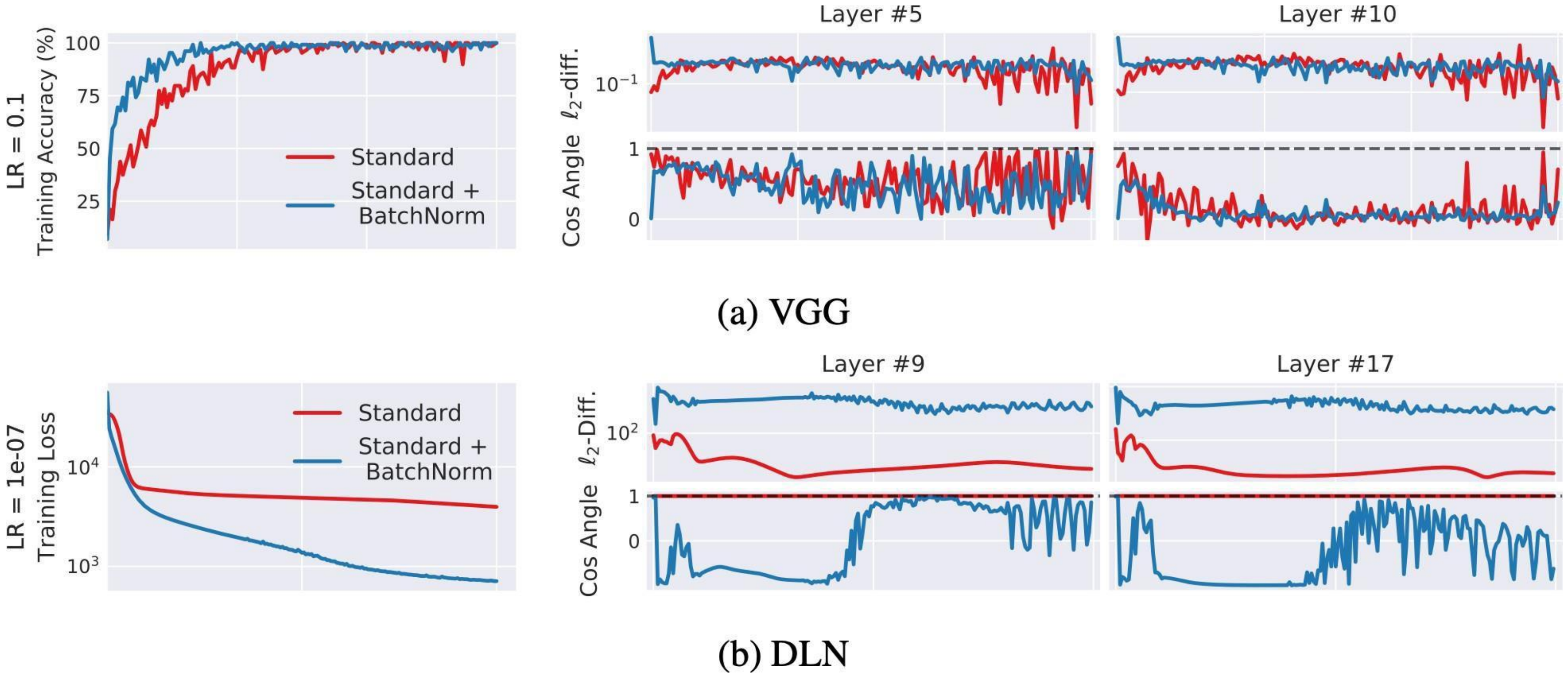
Andrew Ilyas* MIT ailyas@mit.edu

Aleksander Mądry MIT madry@mit.edu

https://arxiv.org/pdf/1805.11604.pdf

Why use BatchNorm?

- What is the relationship between gradients across neighboring iterations?

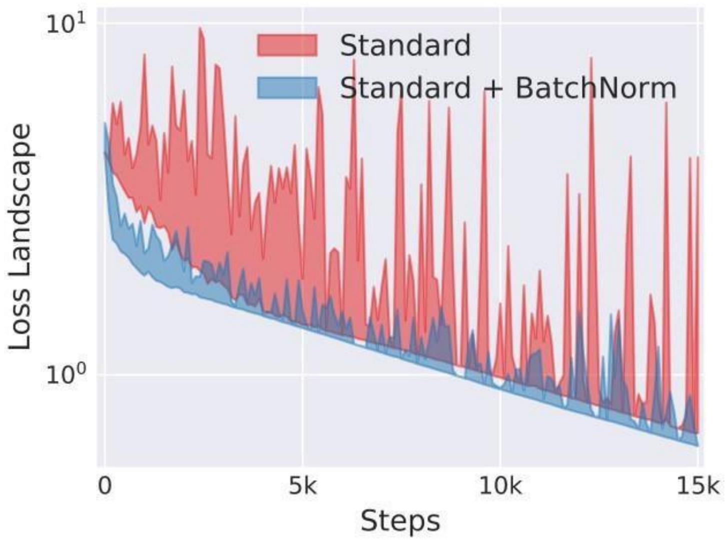


(a) VGG

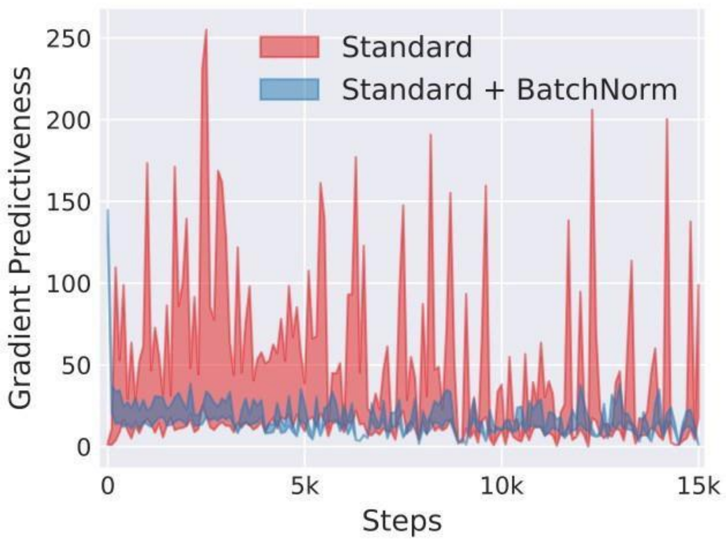
(b) DLN

https://arxiv.org/pdf/1805.11604.pdf

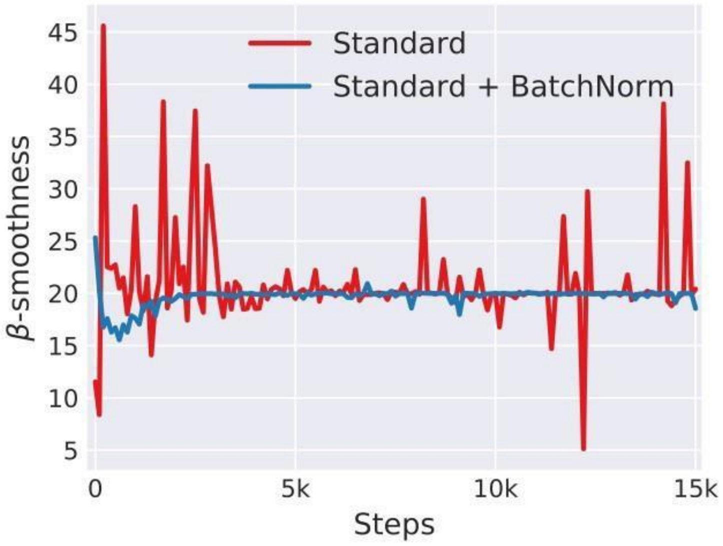
Why use BatchNorm?



(a) loss landscape



(b) gradient predictiveness



(c) "effective" β-smoothness

https://arxiv.org/pdf/1805.11604.pdf

Initialization

Weight initialization

- There should be no symmetries (bad to initialize everything with one number)
- A good option:

$$w \sim \frac{2}{\sqrt{n}} \mathcal{N}(0, 1)$$

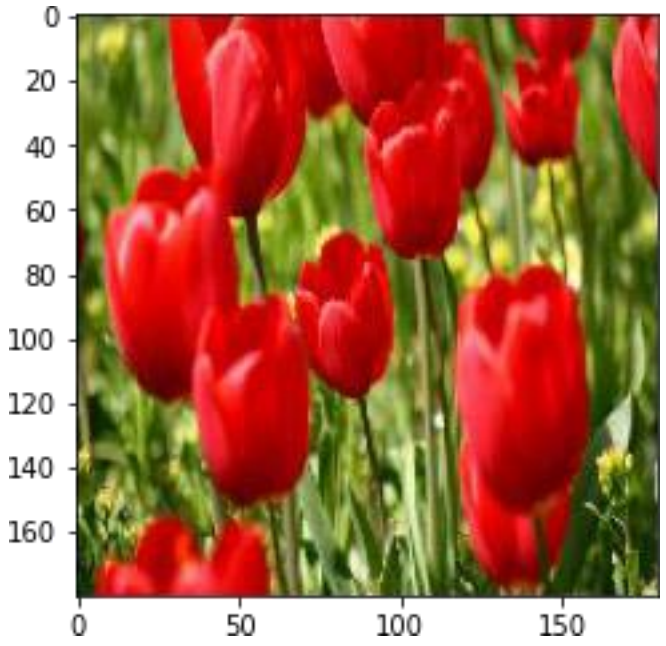
n — layer’s input

- Scale all outputs approximately the same

https://pouannes.github.io/blog/initialization/

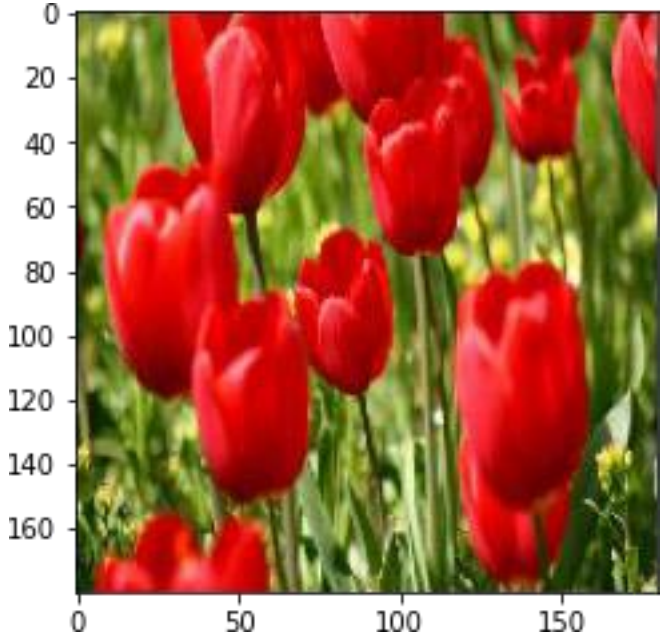
Augmentation

Augmentation



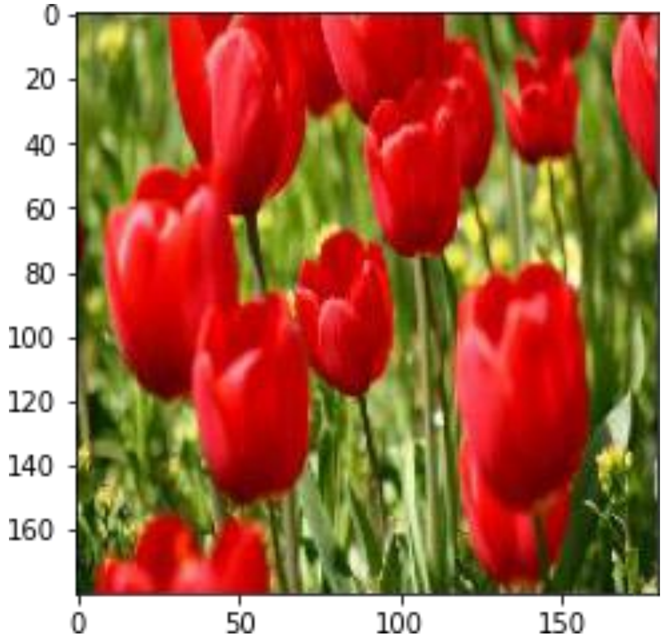
https://www.tensorflow.org/tutorials/images/data_augmentation

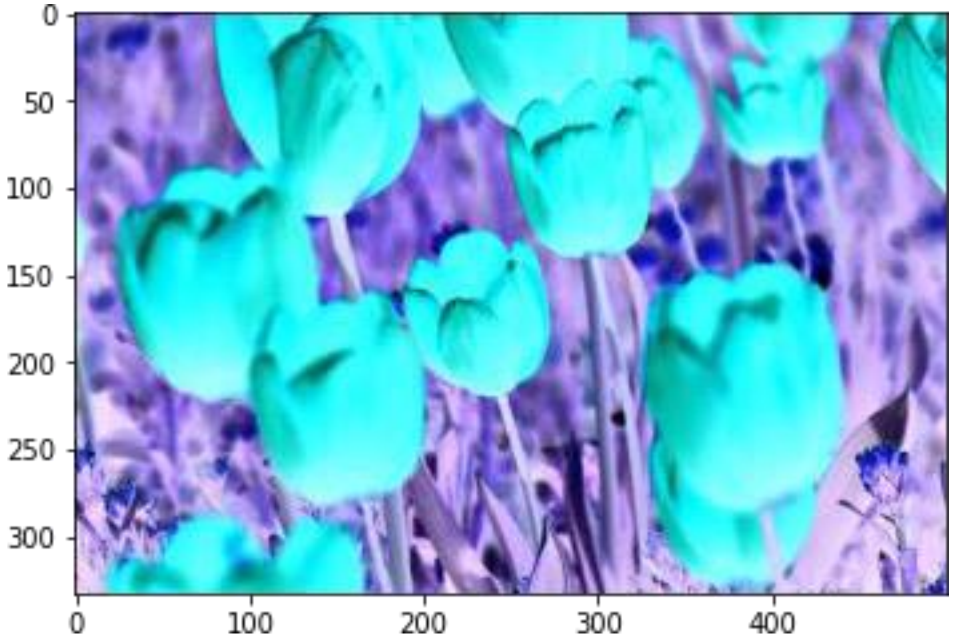
Augmentation



https://www.tensorflow.org/tutorials/images/data_augmentation

Augmentation





https://www.tensorflow.org/tutorials/images/data_augmentation

https://github.com/albumentations-team/albumentations

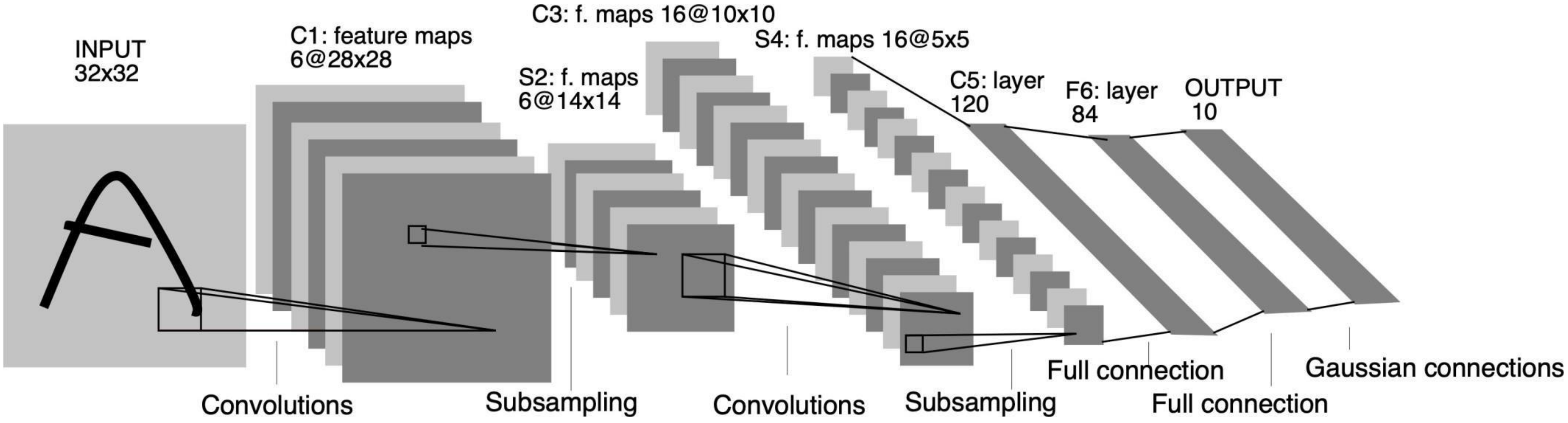
Augmentation

- 'Free' extension of the training dataset
- In some sense, regularization of the model

- Usually, augmentations are randomly applied to images from the current (processing) batch
- During inference, you can apply several augmentations to an image, run the network on each, and average the predictions

Architectures of convolutional neural networks

LeNet (1998)



http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

LeNet (1998)

- MNIST data
- End-to-end learning
- Augmentation
- ~ 60,000 parameters
- Test error rate of 0.8%

http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

ImageNet

- ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

- Approximately 1,000,000 images
- 1000 classes

- Usually, the quality was measured based on the model's best hypothesis

http://image-net.org/challenges/LSVRC/

AlexNet (2012)

ImageNet Classification with Deep Convolutional Neural Networks

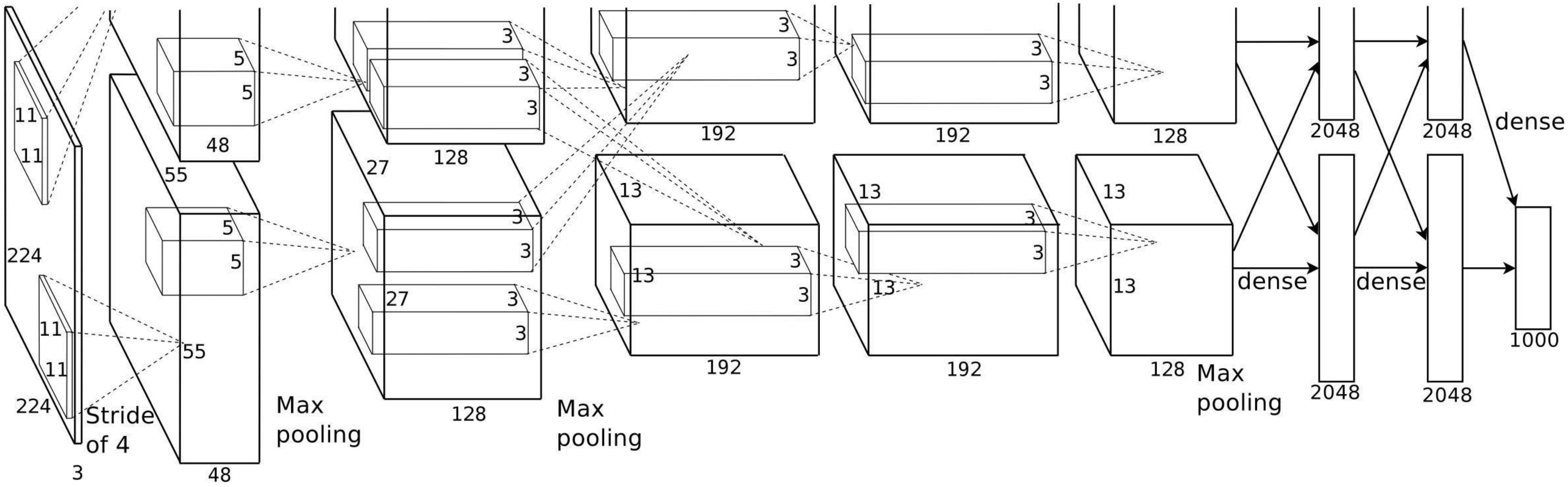
Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

AlexNet (2012)



https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

AlexNet (2012)

- Using ReLU, augmentation, dropout
- Gradient descent with momentum
- Training on two GPUs (5-6 days)
- Approximately 60 million parameters
- Error rate around 17%

https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf